National Physical Laboratory

NPL REPORT MS42

MODEL-BASED SYSTEMS ENGINEERING AT NPL: AN INITIAL INVESTIGATION

ISSUED 1.0

KEITH LINES, DATA SCIENCE
HARISH KRISHNAMURTHY, INSTRUMENTS AND ENGINEERING

FEBRUARY 2023

Model-based systems engineering at NPL: An Initial Investigation

Keith Lines, Data Science
Harish Krishnamurthy, Instruments and Engineering

ABSTRACT

Results are presented of an initial investigation into whether model-based systems engineering (MBSE) [1] and the Systems Modelling Language (SysML) [2] could be of use in the development of cyber-physical systems [3] at NPL. A case study of applying SysML to a generic measurement system is included.

This document is not intended to serve as a detailed introduction to MBSE and SysML. However, sufficient background information is provided to ensure readers unfamiliar with this subject can follow the case study and the authors' conclusions. References for further reading are provided.

DOCUMENT HISTORY

Issue	Date	Author	Change(s)
Issued 1.0	01/02/2023	K Lines	Initial released version.

ISSN 1754-2960

https://doi.org/10.47120/npl.MS42

National Physical Laboratory
Hampton Road, Teddington, Middlesex, TW11 0LW

Approved on behalf of NPLML by
Liam Davies, Group Leader.

CONTENTS

1. INTRODUCTION 1
2. MBSE AND SYSML 1
2.1 WHAT IS MBSE? 1
2.2 WHERE IS MBSE USED? 3
2.3 THREE PILLARS OF MBSE 3
2.3.1 Modelling language 3
2.3.2 Modelling method 5
2.3.3 Modelling tool 5
2.4 IMPLEMENTING MODEL-BASED SYSTEMS ENGINEERING 6
2.5 SUMMARY 6
3. REASONS MBSE COULD BE OF USE TO NPL 6
4. CONCLUSIONS 7
5. FURTHER CASE STUDIES 8
6. ACKNOWLEDGEMENTS 9
7. REFERENCES 9
APPENDIX I: CASE STUDY: GENERIC CALIBRATION SYSTEM 12
AI.1 INTRODUCTION 12
AI.2 SYSML VERSION 12
AI.3 QUALITY MANAGEMENT PLAN 12
AI.4 THE MODEL 13
AI.4.1 Overview 13
AI.4.2 SI Units 16
AI.4.3 User requirements 17
AI.4.4 Functional requirements and use cases 18
AI.4.5 Block definition diagram 21
AI.4.6 Internal block diagram 22
AI.4.7 Further additions 22
APPENDIX II: THE COMPELETE BLOCK DEFINITION DIAGRAM 23
APPENDIX III: IMPORTING THE MODEL INTO PAPYRUS 24
AIII.1 INTRODUCTION 24
AIII.2 INSTALL PAPYRUS 24
AIII.3 INSTALL ECLIPSE MARKETPLACE 24
AIII.4 INSTALL SYSML 1.4 25
AIII.5 UPLOAD THE ARCHIVE FILE 25

1. INTRODUCTION

This report summarises an initial investigation into whether **model-based systems engineering (MBSE)** [1] and the related **Systems Modelling Language (SysML)** [2] could help to specify, design, implement, verify, validate and, above all, **document** complex cyber-physical systems developed at NPL.

SysML provides a formal language and tools to structure the understanding of complexity. Understanding complex systems was identified as a key priority for metrology in the 2030s by the technology and measurement foresighting exercise NPL carried out in 2020 [4]. Awareness and exploration of the latest systems engineering paradigms can only help with that aim.

Complex combinations of hardware and software that interact with their physical surroundings, including occasional input from human operators, i.e. **cyber-physical systems** [3], underpin metrology as they do so for many aspects of the modern world. As will be described in section 3, due to the 2019 redefinition of the SI [5] it is more important than ever for metrology that such systems can be understood, maintained, and further developed **independently of the original and subsequent developers**. The investigation consisted of the following activities:

- Consider some reasons why MBSE could be of use to NPL.
- Apply MBSE to a realistic, but not overly complex, metrology-based example.
- Draw some conclusions.
- If the conclusions are positive, suggest some future case studies.

The report is largely structured as above, beginning with a brief overview of MBSE and SysML.

Table 1 Glossary

Glossary	
Term	**Definition**
INCOSE	International Council on Systems Engineering
MBSE	Model-based systems engineering.
SysML	Systems Modelling Language.
UML	Unified Modelling Language

2. MBSE AND SYSML

Some key concepts of model-based systems engineering and the related Systems Modelling Language are introduced below, along with references for further reading.

2.1 WHAT IS MBSE?

As described by Delligatti [6], model-based systems engineering is best introduced by first considering the more traditional **document-based** approach to systems engineering in which the various stages in the development lifecycle are managed using a disjoint set of spreadsheets, text documents, diagrams and presentations.

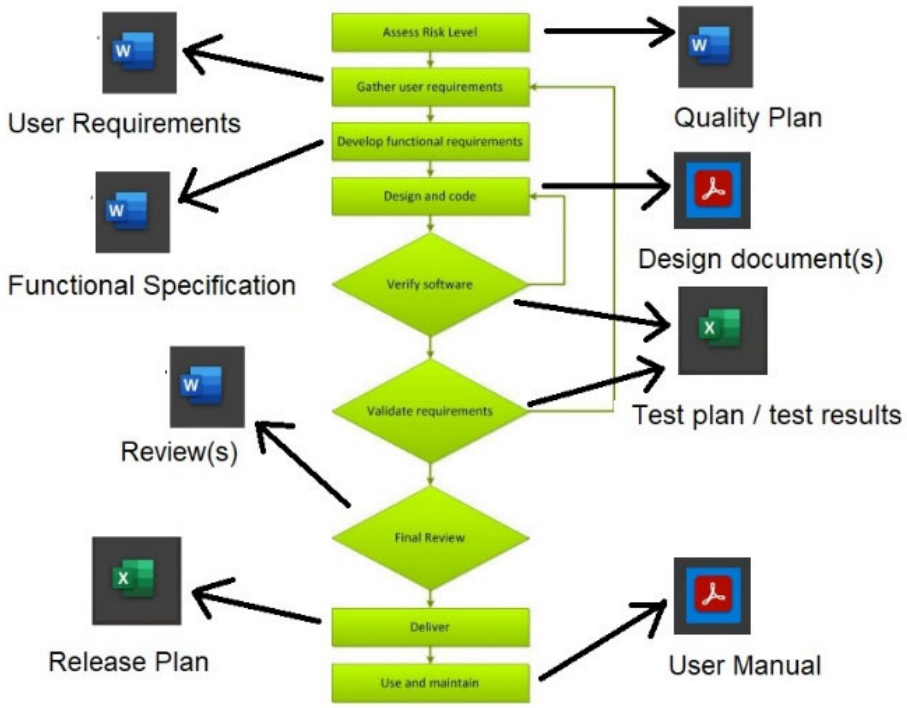

Figure 1 Document-based lifecycle based on a series of non-integrated documents.

The contents of one document will often depend on the contents of another. However, there is no automated connection between documents. The documents may even be subject to different forms of version control.

With MBSEthe set of documents is replaced with a **system model**. The same stages in the development lifecycle are followed and the same set of deliverables are produced, but as Delligatti describes in section 1.1 of [6]:

> "With the MBSE approach, the primary artefact of those activities is an integrated, coherent and consistent system model, created by using a dedicated systems modelling tool. All other artefacts are secondary – automatically generated from the system model using the same modelling tool."

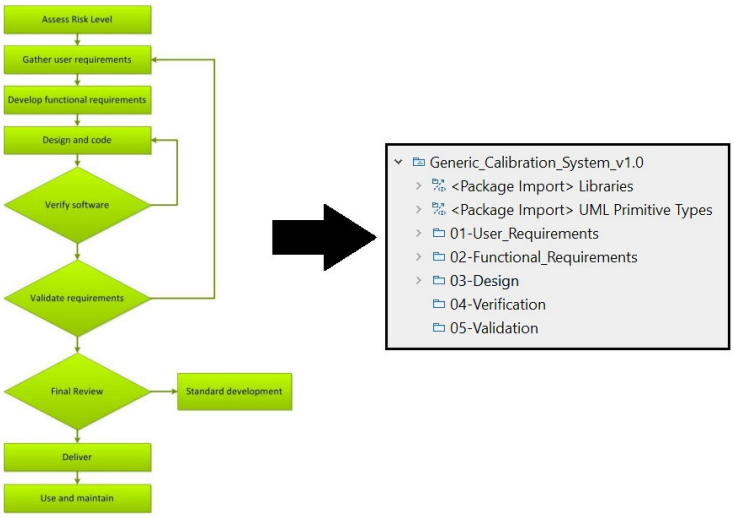

Figure 2 Model-based lifecycle.

The International Council on Systems Engineering (INCOSE) [7] defines MBSE as:

> "…the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases."

INCOSE goes further to say "MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past".

2.2 WHERE IS MBSE USED?

MBSE has been adopted for use in aerospace and transport projects. Case studies are available from organisations such as NASA and Siemens [8, 9, 10].

2.3 THREE PILLARS OF MBSE

Delligatti refers **to three pillars of MBSE**: a modelling language, a modelling method and a modelling tool. These pillars, and pillar-related choices made by the authors, are described below.

2.3.1 Modelling language

A formal language (i.e., not a natural language such as English) is required to create models. The **Systems Modelling Language (SysML)** [2] was chosen by the authors as the modelling language for the case study. SysML is a general-purpose language designed to model both hardware and software systems. Therefore, it was considered an appropriate choice.

SysML is a dialect of the **Unified Modelling Language, version 2 (UML 2)** [11] that customises the language, making it applicable to systems consisting of both hardware and software. UML 2 could have been an appropriate choice for purely software-based systems. Both SysML and UML2 are defined in ISO standards, ISO/IEC 19514:2017 [12] and ISO/IEC 19505-1:2012 [13] / ISO/IEC 19505-2:2012 [14] respectively.

SysML and UML 2 are graphical languages that diagrammatically represent the actors (e.g., system operators, system administrators), use cases, blocks, communication paths etc. of which the system consists. These concepts, and how they are represented in SysML are very briefly introduced using some simple examples.

Use case diagrams

Again, to quote Delligatti:

- Use cases are **the subset of system behaviours** that external actors can directly invoke or participate in.
- An **actor** can be a **person or external system** that interfaces with your system.

SysML represents use cases as ovals placed within a rectangle that represents the system. The usual convention is that human actors are represented using stick figures and external systems are represented using rectangles. Consider the example below:

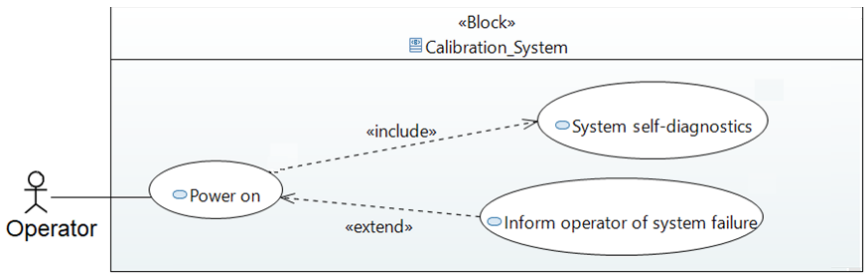

Figure 3 Example use case diagram.

The actor is the system operator and powering up is the system behaviour. The operator initiates this behaviour, and an **included use case** represents the system performing internal diagnostics. Included use cases represent behaviour that could be repeated elsewhere in the system; they are initiated by other use cases, not by an actor.

Extend use cases represent behaviour, that in certain circumstances, is added to another use case. The behaviour in the example is the system raising an error message.

Use cases and use case diagrams can be used to help specify and document functional requirements. They do not directly contain any concept of timing or data flow. Timing can be implied by the order in which use cases are listed in a user case diagram. The case study in APPENDIX I presents use cases ordered in this way.

SysML provides **activity diagrams**, **sequence diagrams** and **state machine diagrams** that directly model a system's dynamic behaviour. These types of diagrams are beyond the scope of this report

Block definition diagrams

Block definitions specify the **types** of entity of which the system will consist. Typically, these entities will be physical items (e.g., computers, artefacts to be calibrated). It is important to understand that block definitions do **not** represent specific **instances** of an entity (e.g., a customer artefact with serial number XYZ359), they represent the **type**.

Readers familiar with object-oriented programming will recognise that a block is like a **class** i.e., a template for creating objects. In fact, SysML blocks are an extension of UML classes. Features such as ports have been added for specifying hardware as well as software.

Block definition diagrams (BDDs) illustrate how these blocks are connected. Consider the following section of a BDD created for the case study:

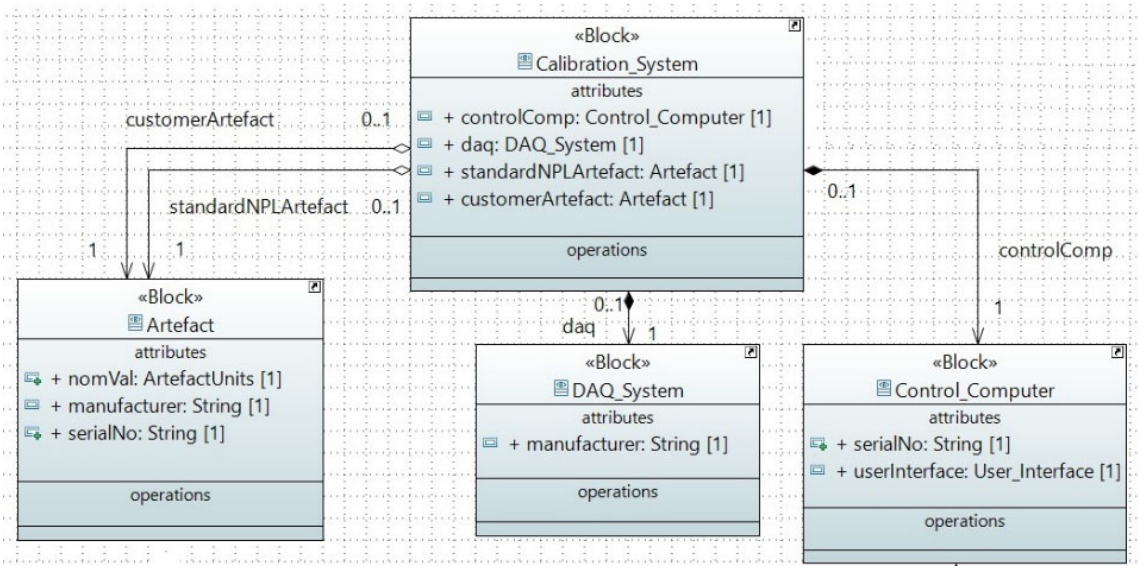

Figure 4: Section of an example block definition diagram.

The block illustrated represents the whole calibration system. Links to other blocks are also illustrated.

- The black diamond at the end of the link indicates a **composite association**. The calibration system is **composed** of one or more data acquisition systems and control computers. These are strong associations; an analogy is that a heart and lungs are in a composite association with a body. Removing the data systems and control computers would disassemble the system.

- The white diamond at the end of the link indicates an **aggregation association**. These are weaker associations than composite. The artefacts have a separate existence from the calibration system. Removing these artefacts would not disassemble either the artefacts or the whole system. Returning to the body analogy something in an aggregation association could be thought of as being like an item of clothing or a pair of glasses.

Each link has a **multiplicity** of 0..1 indicated at the composite end (i.e. near the diamond), where:

- 1 indicates that the calibration system shall contain at most 1 such entity
- 0 indicates the entity can be disconnected from the system.

The calibration system is composed of a customer artefact, an NPL standard artefact, a data acquisition system and a control computer. The links for both artefacts point to the same block because they are both of the same type.

The composition of the calibration system is also indicated in the **attributes** of the **Calibration_System** block. These are values of the types provided by the composite associations.

2.3.2 Modelling method

Perhaps surprisingly, MSBE does not mandate any particular systems development lifecycle. Neither does MBSE state which aspects of the system need to be modelled. The development team shall decide these matters and the **purpose** of the model (e.g. requirements capture / systems design / documentation / all of these?) and how the model will be developed.

For the case study NPL's software development procedure [15] was followed. This procedure provided a good template on which to base the model. Figure 2 illustrates how the iterative development lifecycle specified in [15] maps to an example of a model developed using SysML.

2.3.3 Modelling tool

A variety of software packages for creating and maintaining system models using SysML are available [16]. Some tools must be purchased, others are available free of charge. Some tools are for generic use, and some are intended for a specific purpose such as developing robotic systems [17].

The authors selected the Eclipse Papyrus™ modelling environment [18] for developing the case study. Reasons for selection included:

- Papyrus is free of charge. The expense of purchasing a tool could not be justified for this initial investigation.
- Despite being free, Papyrus is "industry ready" as illustrated by a variety of case-studies on its website.
- It supports the latest versions of SysML and UML.
- It is used within academia for teaching. This aspect was considered appropriate for the authors, who are in the process of becoming familiar with SysML themselves. A British Computer Society colleague of one the authors used Papyrus for teaching undergraduate courses in SysML.

2.4 IMPLEMENTING MODEL-BASED SYSTEMS ENGINEERING

Implementation of MBSE within an organisation is not achieved overnight and requires a natural evolution. The evolution of MBSE within an organisation comprises of the following stages as explained by Holt and Perry in [19]:

1. **Document-based:** A document-based systems engineering process is in place, all the artefacts produced are document based.
2. **Document-centric:** A document-based systems engineering process is in place, but informal or minimal use of UML, SysML is employed.
3. **Model-enhanced:** Modelling is used as part of the overall system engineering process, with an awareness of MBSE and consistent use of modelling languages such as SysML. Documents and models co-exist in this stage.
4. **Model-centric:** MBSE is formalised and consistently applied in multiple projects within the organisation. Most of the artefacts are model based.
5. **Model-based:** In this stage the approach is truly MBSE with mature processes in place. All the artefacts in this stage are purely models.

2.5 SUMMARY

To summarise the above:

- MBSE uses a shared system model as the primary means of information exchange between developers, rather than document-based information exchange. All disciplines, from requirements capture to delivery and maintenance, view a consistent system model.
- The complexity of cyber-physical systems, their surroundings and interdependencies are increasing tremendously in this ever-more connected world. This complexity drives the requirement for repeatable and demonstrable techniques to develop such systems. MBSE offers such capabilities.
- MBSE uses a **formal, digital language** to specify, design, analyse and verify a system. Information is captured in a durable, evolvable format.
- **SysML** is a general-purpose modelling language for systems engineering applications which has emerged as a de facto choice for MBSE. SysML supports a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities.

3. REASONS MBSE COULD BE OF USE TO NPL

The following list summarises the reasons the authors feel that the use of MBSE is worth exploring further in the development of cyber-physical systems at NPL. This list also helped with the conclusions. It is arguable that the first point is the most pressing:

- **Help knowledge capture and transfer**: The 2019 revision of the SI [5] underlined the importance for national metrology institutes of ensuring complex cyber-physical systems can be understood, maintained, and further developed **independently of the original developers**.

  The **Kibble Balance** [20] provides one of the most striking examples of such a system. Prior to the SI revision, the realisation of the kilogram was based on the care and maintenance of a series of physical artefacts. That task was far more complicated and demanding than most could ever appreciate [21]. However, the realisation of this

fundamental unit of measurement now requires an in-depth understanding of a highly complex combination of hardware and software that is the result of many person-decades of development.

Such systems must be maintained, and the next generation developed, **without the need to contact previous developers.** MBSE could ease the task of ensuring key information has been thoroughly captured and can be passed on to future generations of developers / maintainers.

- **Reuse of designs:** MBSE provides the ability to arrange institutional knowledge using a formal language, allowing for reuse and broad exposure.

- **Keeping documentation thorough and consistent:** Maintaining a comprehensive set of quality plans, requirements documents, design documents, test plans etc. can be a complicated, tedious, and error-prone task. Changes to one document (e.g., the addition of a new user requirement), will almost certainly have to filter through to other documents (e.g., new functional requirements, modifications to design documents and new test cases).

  A single, consistent, unambiguous system representation ensures integrity and traceability throughout the development lifecycle and maintenance. MBSE provides an alternative to a set of unlinked documents. Changes to one stage of system specification or development can automatically filter through to other stages.

  Focusing on information integration rather than document generation allows for detection of inconsistency / staleness.

- **Gap analysis:** “Retro-fitting” MBSE to existing systems is not ideal. However, it is the experience of the authors that reverse engineering software quality management documentation is always a worthwhile exercise in gap analysis (e.g., ensuring test plans are sufficiently thorough). The authors wanted to explore whether the same could be said of MBSE.

- **Greater engagement:** Could MBSE be technically more interesting than traditional document-based approach to systems engineering? Anything that would help engage scientists / engineers with what might otherwise be considered burdensome bureaucracy must be worth investigating.

4. CONCLUSIONS

The following are the conclusions of this work, in no order of priority:

- **Some features of MBSE and SysML are immediately applicable to NPL projects:** Referring to the stages of introducing MBSE, listed above, **stage 2 document-centric**, could be the most realistically obtainable (or even desirable) stage of introducing MBSE within NPL. Reasons include:
  - MBSE and SysML is complex with a non-trivial learning curve. Publicly available case studies and papers, such as those referred to in section 2.2, indicate that MBSE is mainly applied to large-scale aerospace and defence projects. Therefore, MBSE using SysML may appear to be overkill for most NPL’s projects.

- However, there are **features** of MBSE and SysML that would be of practical benefit to some projects. E.g., **use cases** and **block definition diagrams** provide a useful, practical tool for requirements capture and systems design. SysML provides **formality** and therefore consistency from one developer to another. The resulting systems would be better designed and **better documented**.

  The skills required for those features should be well within the grasp of any competent coder and already second nature to any software or systems engineer.

- **Invest in a good-quality modelling tool:** Following on from the above conclusion, NPL would need to invest in a good-quality modelling tool. Software tools purely for drawing diagrams, e.g. Microsoft Visio, would not be sufficient. It is essential that the tool enforces the syntax / semantics of the language to ensure consistency.

  Papyrus has done an excellent job for the case study. However, NPL would need features that Papyrus does not provide, such as the ability to embed mathematical equations in requirements which may mean purchasing a tool.

- **Useful for engaging with external partners:** MBSE is used by industrial partners with whom NPL collaborates (e.g., within the aerospace sector [22, 23]). A knowledge of MBSE could help with communication with developers within these organisations.

- **Work closely with Instrumentation and Engineering:** There is a major internal change program within NPL's Instrumentation and Engineering department. The aim is to introduce formal systems engineering within NPL. Referring to section 2.4, NPL generally is not even in stage 1 of the evolution of MBSE.

- **MBSE using SysML is <u>fun</u>. That's something not to be underestimated:** As was anticipated in section 3, developing the case study was an engaging and enjoyable exercise. Such an approach really does have the potential to make vital, but far too often neglected, tasks more engaging. Experience of MBSE and SysML will boost the CVs of early-career members of staff. Also, the opportunity for staff to develop in this area may reduce attrition as well as encouraging a "quality first" mindset.

- **External support required:** In the longer term it is recommended to seek external support (which could prove very expensive) or train someone to gain MBSE expertise.

- **OVERALL CONCLUSION:** MBSE is a discipline for understanding, managing and "taming" complex systems that is being widely adopted within industry. Its application to areas in which NPL are active (such as Industry 4.0 [24], digital twins [25] or understanding complex systems as a priority for metrology in the 2030s [4]) implies that at the very least NPL **should have an awareness** of MBSE. Judiciously applying MBSE to NPL projects will aid that awareness.

5. FURTHER CASE STUDIES

This work can be progressed by applying SysML to further, more realistic, examples. NPL provides numerous examples of cyber-physical systems for MBSE case studies. The most practical approach would be to model selected aspects of a system rather than the whole system. Beginning by developing use cases of operator interaction would be an excellent and challenging starting point. The resulting documentation would aid understanding by those new to the system. It may also help optimise design Possible examples include:

- **Cryogenic Current Comparator Bridge (CCC Bridge):** CCC Bridges are used for the calibration of standard resistors against Quantum Hall Resistance (QHR) standards [26]. The CCC Bridges were used as a basis for the Generic Calibration System presented in

APPENDIX I. Extending the case study presented in this report to include features of a CCC Bridge could help create a more “realistic” model.

- **Business Systems:** MBSE can help with the development of business, as well as scientific, systems of which NPL provides numerous examples. In this case the Business Process Model Notation (BPMN) [27] is used rather than SysML. However, BPMN is also derived from UML and the shared origin with SysML means they complement each other. For example, in [28] Ćwikła et al. present an outline of a case study that combines scientific and business aspects. Both SysML and BPMN are used to model a system that collects and processes real-time data from the monitoring of a production system. This data is then used for the purposes of company management.

  BPMN has also been defined in an ISO standard, ISO/IEC 19510:2013 [29].

In [30] Hein et al. present an example of a detailed case study. An NPL equivalent of such a case study would have great value for reasons described in the conclusion.

6. ACKNOWLEDGEMENTS

This work was undertaken jointly by the National Physical Laboratory’s Data Science and Instruments teams as part of Data Science’s Tools for Trustworthiness National Measurement System (NMS) project 2021 – 2022 and 2022 – 2023.

Louise Wright provided helpful comments on an early draft of this report. Figures 1 and 2 incorporate a flowchart from [15], the authors thank the NPL Quality Assurance team for permitting it to be used. Thanks also to Peter Harris and to Ian Robinson and Rob Smith for helpful comments.

7. REFERENCES

1. Model Based Systems Engineering (MBSE). Retrieved 05/12/2022 from International Council on Systems Engineering (INCOSE): https://www.incosewiki.info/Model_Based_Systems_Engineering/index.php?title=MBSE_Definitions
2. Systems Modelling Language homepage. Retrieved 05/12/2022 from SysML.org: https://sysml.org/
3. National Institute of Standards and Technology (NIST), Cyber-physical systems Topics. Retrieved 05/12/2022 from NIST: https://www.nist.gov/cyberphysical-systems
4. Technology and Measurement Foresighting: A vision of the 2030s shaped by metrology. Retrieved 05/12/2022 from NPL: https://www.npl.co.uk/foresighting
5. SI redefinition. Retrieved 05/12/2022 from the International Bureau of Weights and Measures (BIPM): https://www.bipm.org/en/measurement-units
6. Delligatti, L. SysML Distilled: A brief guide to the system modelling language, Published by Pearson Educational, 2014, ISBN-13 978-0-321-92786-6
7. International Council on Systems Engineering (INCOSE). Retrieved 05/12/2022 from INCOSE: https://www.incose.org
8. Bonanne, K H. Model-based systems engineering for capturing mission architecture system processes with an application case study: Orion Flight Test 1. Retrieved 05/12/2022 from NASA: https://trs.jpl.nasa.gov/handle/2014/43714

9. Model Based Systems Engineering (MBSE). Retrieved 05/12/2022 from Siemens: https://www.plm.automation.siemens.com/global/en/our-story/glossary/what-is-model-based-systems-engineering/28573
10. Roodt, D; Nadeem, N; Vu, L. Model-Based Systems Engineering for complex rail transport systems – A case study. Retrieved 05/12/2022 from Shoal Group: https://www.shoalgroup.com/wp-content/uploads/2020/08/IS_2020_Papers_paper_151-1.pdf-1.pdf
11. Unified Modelling Language (UML) homepage. Retrieved 05/12/2022 from OMG: https://www.omg.org/spec/UML
12. ISO/IEC 19514:2017: Information technology — Object management group systems modeling language (OMG SysML). Retrieved 05/12/2022 from ISO: https://www.iso.org/standard/65231.html
13. ISO/IEC 19505-1:2012: Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 1: Infrastructure. Retrieved 05/12/2022 from ISO: https://www.iso.org/standard/32624.html
14. ISO/IEC 19505-2:2012: Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure. Retrieved 05/12/2022 from ISO: https://www.iso.org/standard/52854.html
15. NPL internal quality procedure: QPNPL/M/013 NPL-Written Software
16. Free & Commercial SysML Tools for MBSE. Retrieved 05/12/2022 from SysML Tools https://sysmltools.com/
17. RoboTool. Retrieved 05/12/2022 from the University of York. https://robostar.cs.york.ac.uk/robotool/
18. Eclipse Papyrus Modelling environment. Retrieved 05/12/2022 from Eclipse: https://www.eclipse.org/papyrus/
19. Holt, J; Perry, S. Implementing MBSE into your Business: The Trinity Approach, Published by INCOSE UK Limited, ISBN 978-0-9934857-5-6
20. Robinson, I. The architecture of the NPL next generation Kibble balance. TO BE PUBLISHED.
21. Reynolds, M. This is why physicists just completely redefined the kilogram. Retrieved - 05/12/2022 from Wired: https://www.wired.co.uk/article/how-many-grams-in-a-kilogram-weight-change-planck-constant
22. TRUTHS satellite calibration mission. Retrieved 05/12/2022 from NPL: https://www.npl.co.uk/earth-observation/truths
23. ESA awards contract to Airbus UK for TRUTHS predevelopment. Retrieved 05/12/2022 from the European Space Agency: https://www.esa.int/Applications/Observing_the_Earth/ESA_awards_contract_to_Airbus_UK_for_TRUTHS_predevelopment
24. Digital Transformation Monitor Germany: Industrie 4.0. Downloaded 05/12/2022 from Europa.eu https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%204.0_DE.pdf
25. Wilking F, Sauer C, Schleich B, Wartzack S. Sysml 4 Digital Twins – Utilization of System Models for the Design and Operation of Digital Twins. Retrieved 05/12/2022 from Cambridge University Press: https://www.cambridge.org/core/journals/proceedings-of-

the-design-society/article/sysml-4-digital-twins-utilization-of-system-models-for-the-design-and-operation-of-digital-twins/2B934D8367F74EA1E2CB6789096EA268

26. Kleinschmidt P, Williams J, Fletcher N, Janssen JT. Cryogenic Current Comparator for Quantum Hall Resistance Ratio Measurements. Retrieved 09/10/2022 from NPL: http://eprintspublications.npl.co.uk/2168/1/bemc2001-11.pdf
27. Object Management Group Business Process Model and Notation. Retrieved 05/12/2022 frombpmn.org: https://www.bpmn.org/
28. *G Ćwikła et al 2017 IOP Conf. Ser.: Mater. Sci. Eng.* ***227*** *012034*. Analysis of the possibility of SysML and BPMN application in formal data acquisition system description. Retrieved 05/12/2022 from the Institute of Physics: https://iopscience.iop.org/article/10.1088/1757-899X/227/1/012034
29. ISO/IEC 19510:2013: Information technology — Object Management Group Business Process Model and Notation. Retrieved 05/12/2022 from ISO: https://www.iso.org/standard/62652.html
30. Hein A, Karban R, Weilkiens T, Zamparelli M. Cookbook for MBSE with SysML. Retrieved 05/12/2022 from ResearchGate: https://www.researchgate.net/publication/268977905_Cookbook_for_MBSE_with_SysML
31. About the OMG system modeling language specification version 1.4. Retrieved 08/10/2022 from OMG: https://www.omg.org/spec/SysML/1.4/About-SysML
32. SI Brochure: The International System of Units (SI) version 9th edition 2019. Downloaded 05/12/2022 from BIPM: https://www.bipm.org/en/publications/si-brochure
33. [SysML] SI Units Library & Import (Is the library included and how to import it?) Downloaded 05/12/2022 from Eclipse Community Forums: https://www.eclipse.org/forums/index.php/t/261774/

APPENDIX I: CASE STUDY: GENERIC CALIBRATION SYSTEM

AI.1 INTRODUCTION

This appendix presents an overview of the case study undertaken as part of this initial investigation. For access to the Papyrus files contact keith.lines@npl.co.uk. Alternatively, for NPL staff these files can be found in the Model Based Systems Engineering Office 365 Team.

The original intention was to apply MBSE to a "real" example from NPL. It soon became clear that such a complex case study would have been too ambitious at this initial stage. Therefore, a less detailed **generic calibration system** was specified instead. The system calibrates "customer artefacts" against "NPL standard artefacts" and does not specify the details of the artefacts themselves (they could be resistors, masses, hydrophones, length bars etc.). This case study draws on the authors' experience of developing, maintaining, documenting and project-managing cyber-physical systems at NPL. Therefore, although generic, it is sufficiently realistic to make the exercise worthwhile.

The overview begins by summarising the quality management plan that provided the starting point for developing the model. The elements of the model, which are structured according to the quality management plan are then described. Those quality plan elements considered in this case study are user requirements, functional requirements and system design.

AI.2 SYSML VERSION

SysML version 1.4 [31] was used for this case study. Newer versions of the language have been defined. However, it was decided to use a slightly more mature version in the hope that more tools, examples, background reading etc. would be available than for newer versions.

Details of the versions of the language can be found on the Systems Modelling Language homepage [2].

AI.3 QUALITY MANAGEMENT PLAN

NPL's software quality management system [15] was used as the basis for developing the model. Therefore, work begun by generating a documented **quality management plan** using an MS-Word template provided by the quality system. A summary of the plan is listed below.

Table AI.1 Quality management plan

Brief description	• A case study to help determine whether the Systems Modelling Language (SysML, https://sysml.org/) would be of use at NPL for developing hardware and software.
	• The case study shall be sufficiently realistic and detailed to draw some useful conclusions. However, it should not be so detailed as to require previous experience with SysML.
	• The specification shall contain components common to all measurement systems; i.e.:
	  ○ A standard artefact against which other artefacts are calibrated.
	  ○ The artefact to be calibrated.
	  ○ Data Acquisition (DAQ) hardware and software.
	  ○ A data storage system where measurement data will be held.

	○ A control computer running software for managing the calibration process.
User requirements	Documented user requirements Review by team Review by suitably qualified independent person Review by customer or proxy
Functional requirements	Documented functional requirements Traceable requirements Review by team Review by suitably qualified independent person
Design	Clear and well-structured design Tool support Review by team Review by suitably qualified independent person
Verification	Module testing as coding progresses Effective testing of complete software against specification Review by team
Validation	Documented testing against specification Review by team Review by suitably qualified independent person
Delivery, use and maintenance	Version control on release Version control before release Bug tracking / error logging Traceability User documentation

It may seem contradictory to begin a model-based case study using a document. However, the plan provided excellent guidance for this case study as to what the model should contain. It MBSE is adopted then perhaps a number of templates for models could be an alternative to document-based quality plans.

AI.3.1 Reviews and customer approval using MBSE

An important point to consider is **how would reviews be carried out, and recorded, using an MBSE model rather than documents?** Expertise in MBSE should **not** be necessary to carry out reviews. Customer approval would also need to be addressed. A naïve approach would be that the user requirements can be grouped together into a “view” that could be reviewed (that view could be a document).

Consultation with MBSE experts, perhaps via INCOSE [7], may be necessary to address this point. It is likely that the tool used generate and update the model will provide review facilities.

AI.4 THE MODEL

AI.4.1 Overview

The structure of the model follows the structure of the quality management plan. There is one **package** for each of the plan requirements. A SysML package can be thought of as a folder, used to group together **model elements**.

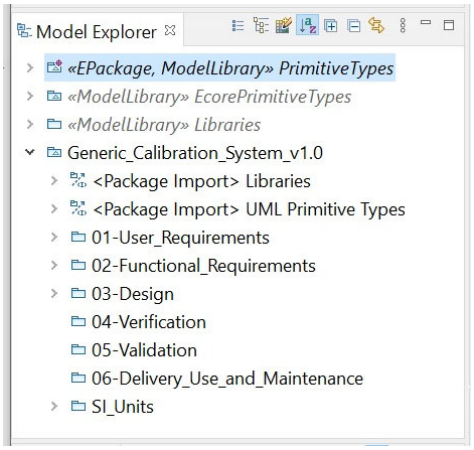

Figure AI.1 Model structure.

The interface Papyrus provides for developing MBSE models allows the user to create and browse **projects**, **models** within the projects, the **packages** within the models and the **elements** within the packages. Figure AI.2 provides an example screen dump showing block definitions.

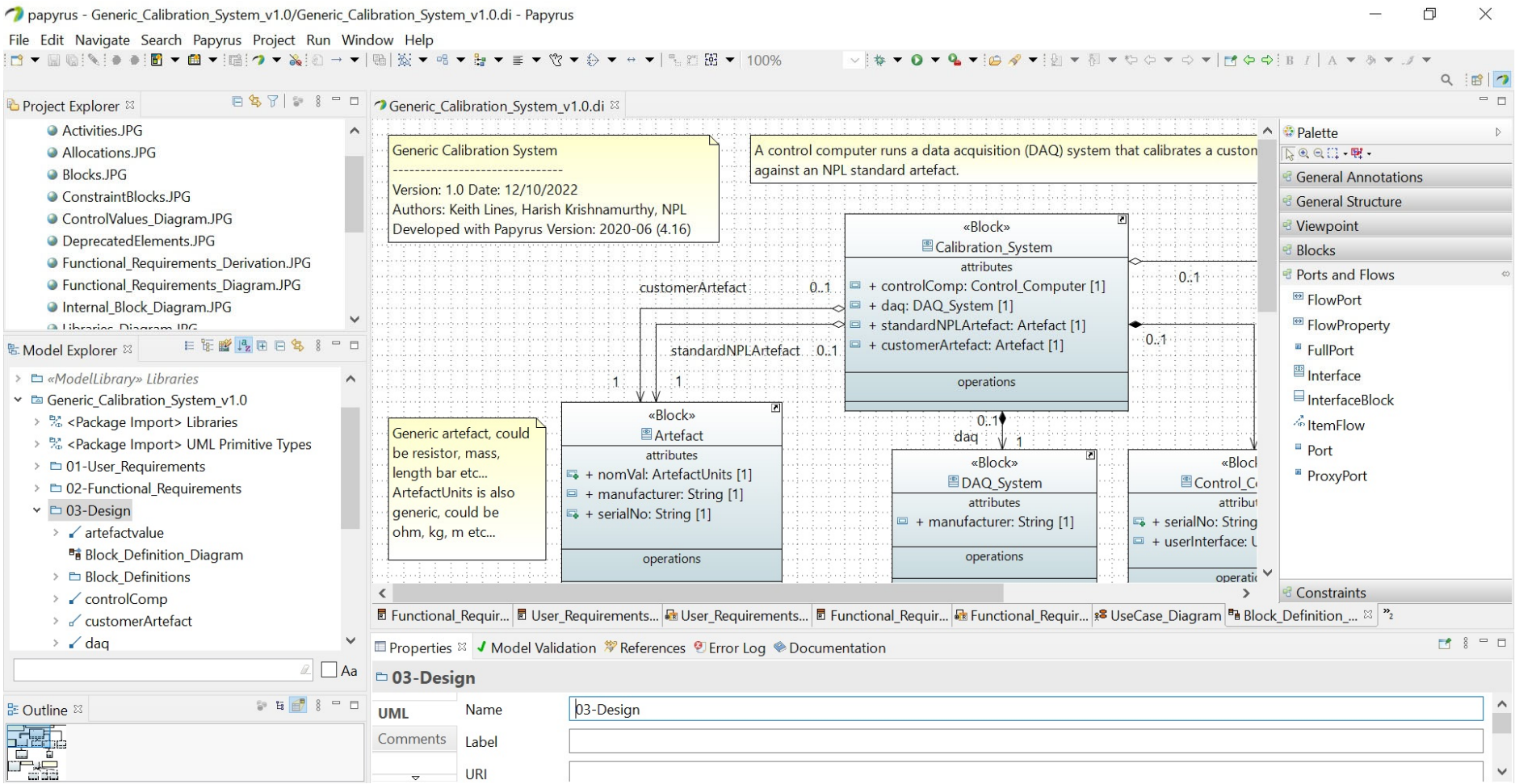

Figure AI.2 Papyrus Project Explorer.

AI.4.2 SI Units

Before providing an overview of the model as a whole, the contents of the **SI_Units** package is described. This package provides **value types**, **units** and **quantity kinds** defined as closely as possible to the SI Brochure version 9 [31]. It is beyond the scope of this document to describe concepts from [31] (e.g., base quantities, derived quantities, base units and derived units) in detail.

SysML provides **value types**, in addition to the usual primitive types (Boolean, Integer, Real, String and enumerated values), which allow values to be assigned types that more accurately describe the system being developed than primitive types. E.g., in the diagram below the nominal values of the mass artefacts are expressed in kilograms, rather than Integer or Real.

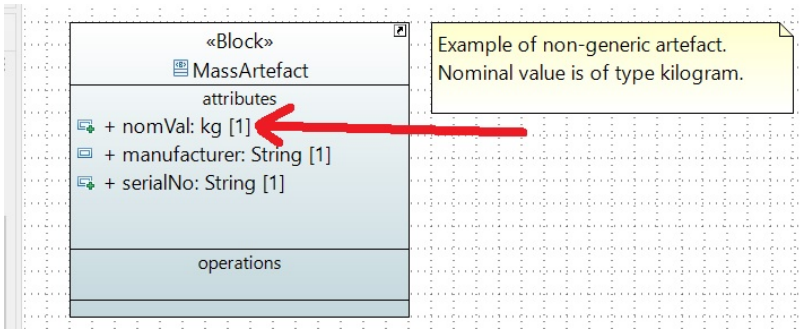

Figure AI.3 Example use of value type.

The **kg** value type is defined in the model as a **specialization** (i.e., a subclass) of the **Real** value type. The arrow in the diagram below defines a **generalization** between kg and Real. Generalization can be thought of as the inverse of specialization, with the arrow pointing in the opposite direction.

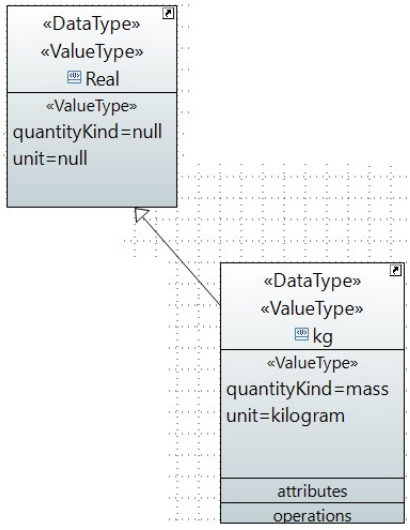

Figure AI.4 Example use of value type.

As illustrated, value type kg is assigned a **quantity kind** of **mass** and **unit** of **kilogram**. In SysML quantity kinds are equivalent to the **base** and **derived quantities** defined in [31]. The seven base quantities are time, length, mass, electric current, thermodynamic temperature, amount of substance and luminous intensity. Derived quantities include absorbed dose, capacitance, electric potential difference and electric resistance.

The package **SI → 02-Base Quantities** contains quantity kinds for each of the base units.

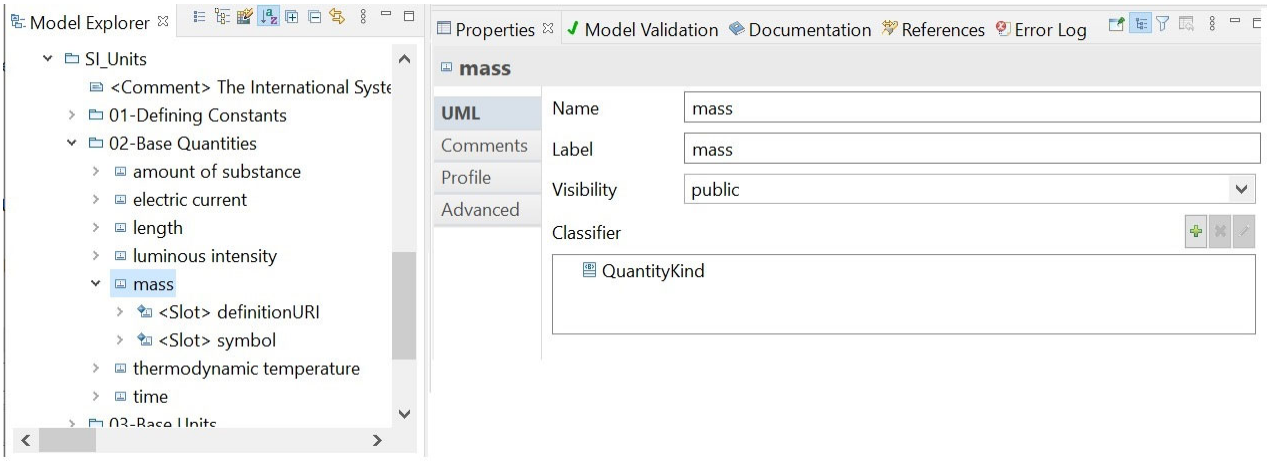

Figure AI.5 Quantity kinds defining base quantities.

The seven base SI units are second, metre, kilogram, ampere, kelvin, mole and candela. Derived quantities include gray, farad, volt and ohm.

The package **SI_Units → 02-Base Units** contains **unit definitions** for each of the base units. This package also contains **value type definitions** for each of the base units. As illustrated below, the value type kg has associated unit kilogram and quantity kind mass.

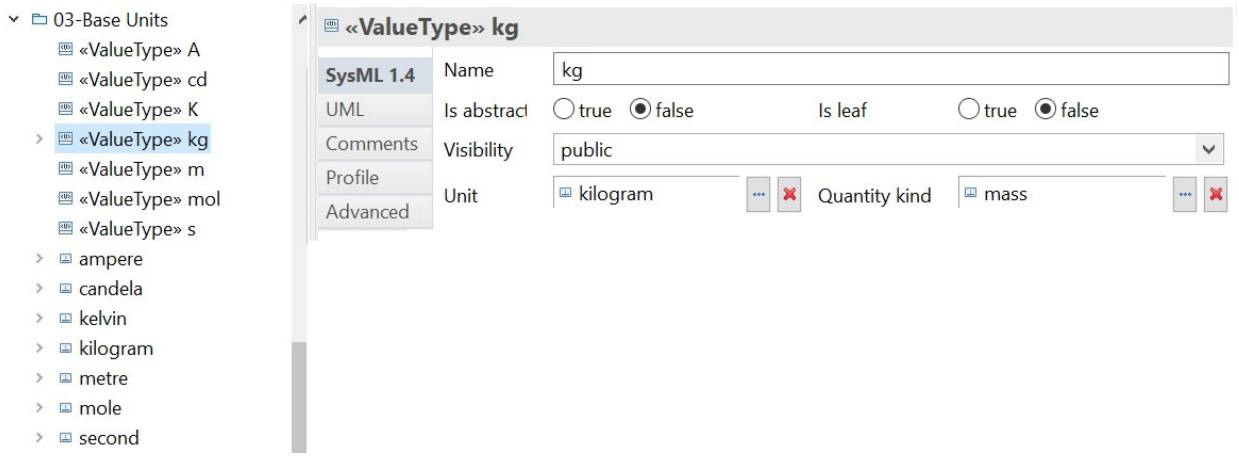

Figure AI.6 Base units and value types.

The SI_Units package contains definitions for SI quantities and units and the constants that as from 2019 define the SI [5].

Pre-built packages containing SI units are available for the various SysML modelling tools [32]. However, developing this SI_Units package was a useful exercise in learning SysML. It is also unlikely that any pre-built packages will have yet taken the 2019 redefinition into account.

AI.4.3 User requirements

As few and as high-level user requirements as possible were specified. Four user requirements were defined. In the Papyrus model explorer, expanding package **01-User_Requrements** displays these requirements as individual objects within the model.

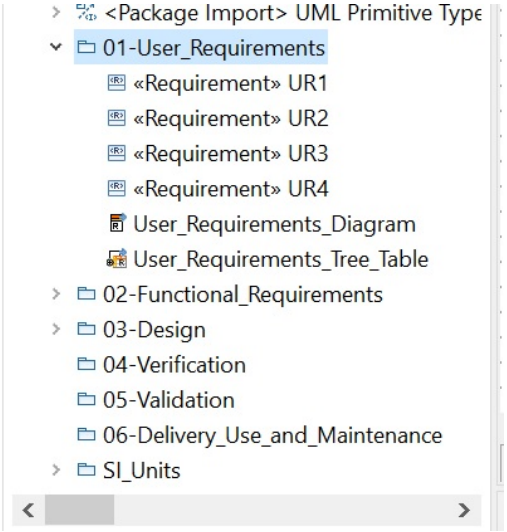

Figure AI.7 User requirements package.

The user requirements diagram brings them together:

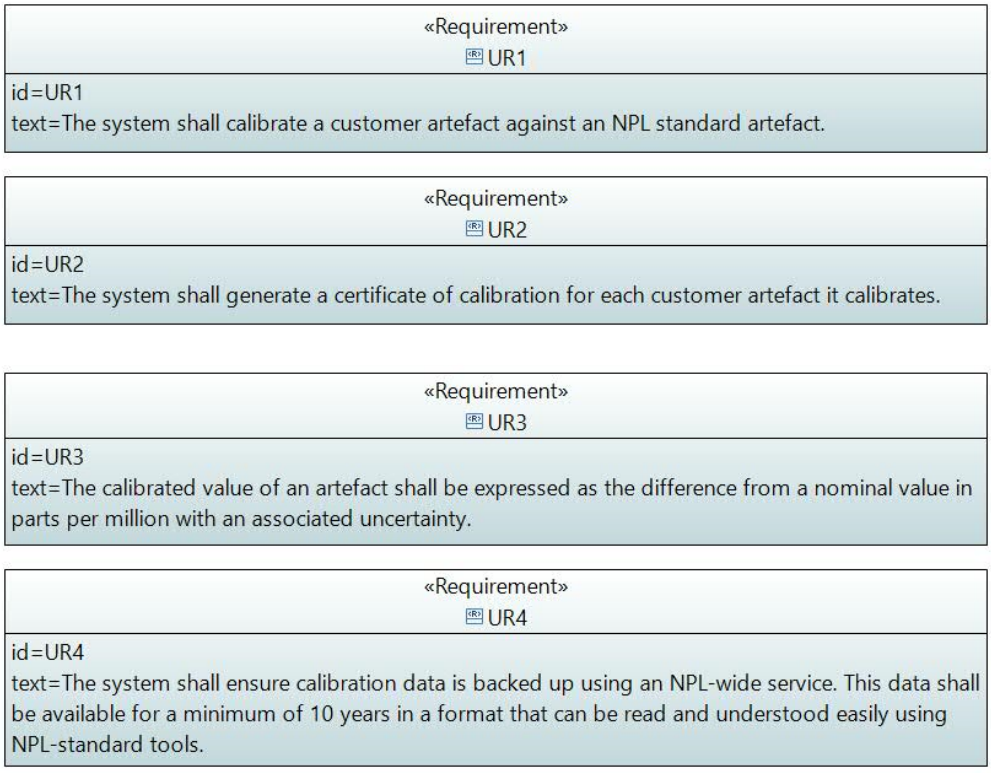

Figure AI.8 User requirements diagram.

AI.4.4 Functional requirements and use cases

Having determined the user requirements, the next stage (according to the quality procedure being followed [15]) is to map them to functional requirements. Functional requirements will determine **what** is to be implemented not **how**. They should be understandable by someone who cares about how the system works but is not a coder or systems developer.

The package **02-Functional_Requirements** contains the functional requirements, mappings from the user requirements and a package containing the **use cases**. As use cases help define the functionality of the system, it was decided appropriate to include use cases as a sub-package within the functional requirements:

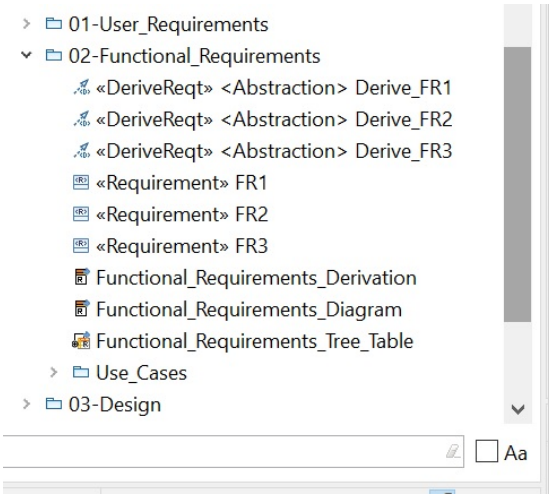

Figure AI.9 Functional requirements package.

The following section of the functional requirements diagram list the first three requirements.

«Requirement» FR1
id=FR1 text=The system shall make a number of measurements of the customer artefact, against a NPL-standard artefact. The system shall provide a means of grouping these measurements together to allow calibration values to be calculated.

«Requirement» FR2
id=FR2 text=The system shall allow an operator to reject customer artefact measurements. The operator must enter a reason for rejection.

«Requirement» FR3
id=FR3 text=The system shall calculate the calibration value of a customer artefact from a series of grouped-together measurements that have not been rejected.

Figure AI.10 Section of functional requirements diagram.

The **Functional_Requirements_Derivation** diagram illustrates the traceablity between user and functional requirements. The following section of the diagram, illustrates how functional requirements FR1 to FR3 are tracable to user requirement UR1. This traceablity is essential to validating that the user requirements have been met.

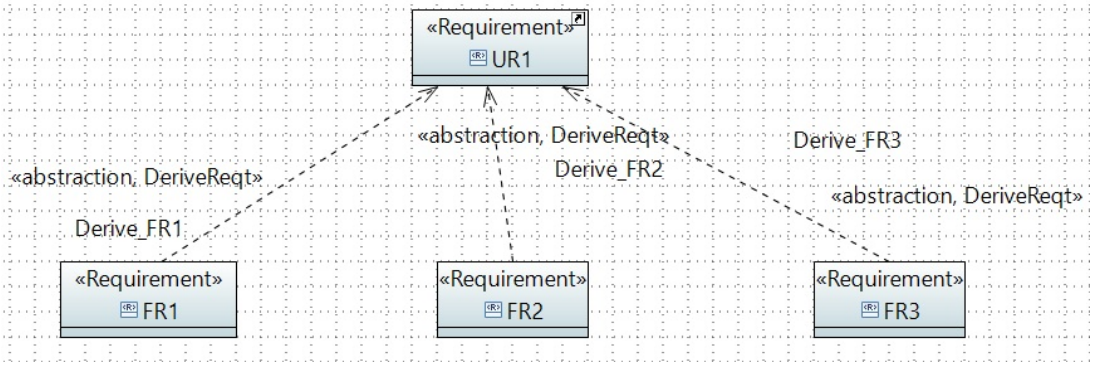

Figure AI.11 Requirements traceability.

The **use cases** are now considered, they are summarised using the diagram provided below.

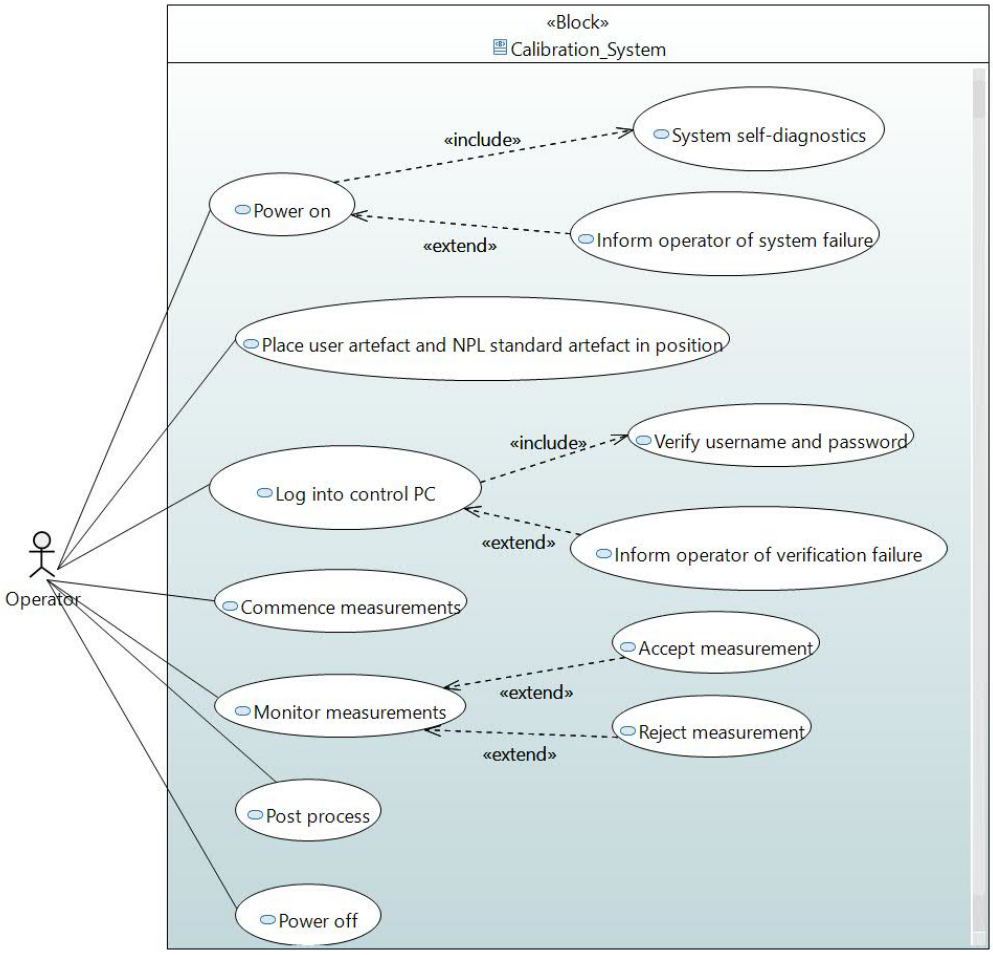

Figure AI.12 Use case diagram.

The operator, represented using a stick figure, is the one external actor modelled with this system. The use cases defined for this system are obviously very high-level, e.g. “Commence measurements” encapsulates a lot of detail that would need to be made explicit in a more realistic case study.

Use cases contain no explicit details of timing. The sequence in which the operator’s actions take place is inferred by the ordering of the use cases in the diagram. Within the **Use_Cases**

package the use cases are organised into “phases”. It should also be noted how use cases can be traced to functional requirements:

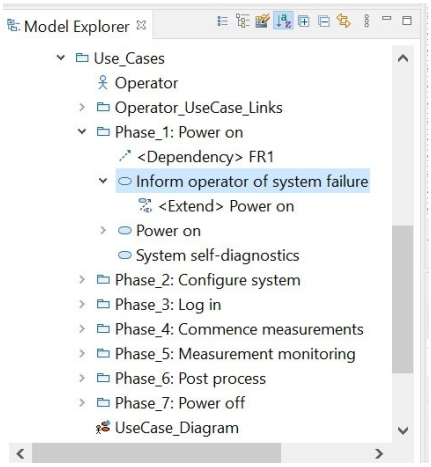

Figure AI.13 Use_Cases package.

AI.4.5 Block definition diagram

The **block definition diagram** is the heart of the SysML model. These blocks represent the **types of elements** from which the system will ultimately be constructed. An example section of the block definition diagram of the generic measurement system is provided below:

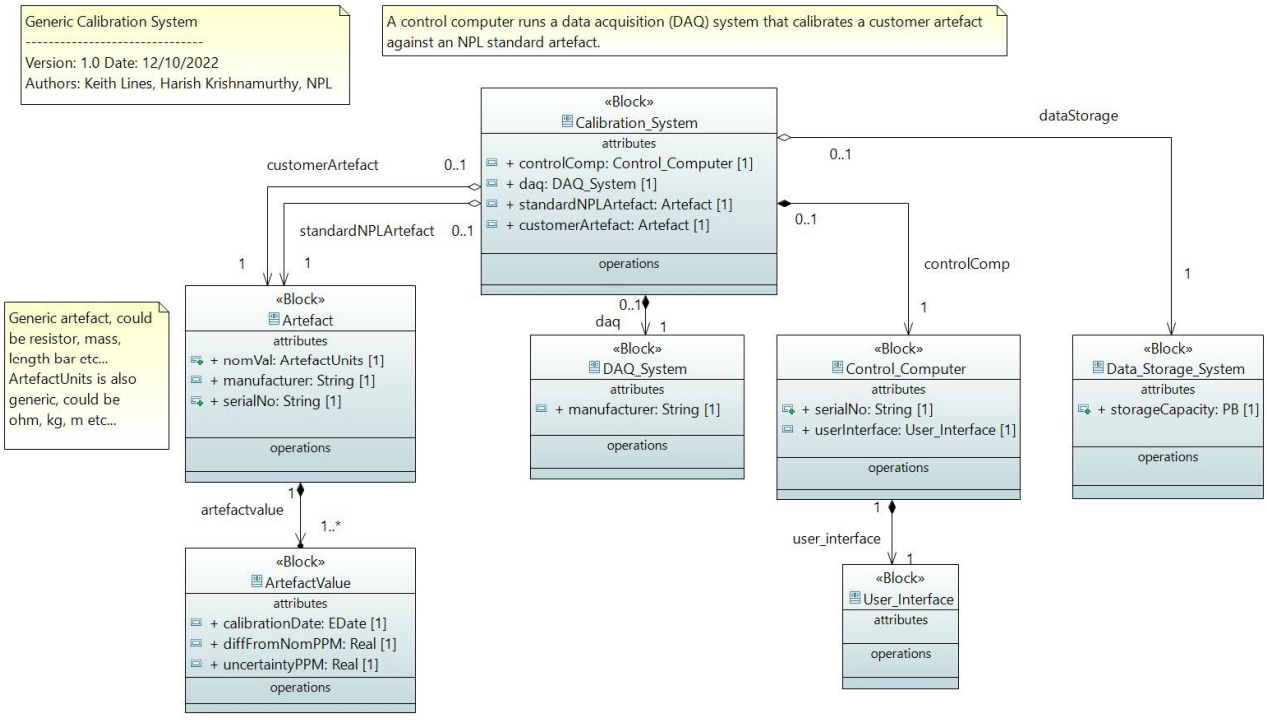

Figure AI.14 A section of the block definition diagram.

The diagram illustrates the relationships between the blocks. For example, although the nominal value of an artefact will remain unchanged, its actual value is highly likely to vary over its lifetime. Accordingly, the **Artefact** block is linked to the **ArtefactValue** block using an **association** (i.e., link) with a multiplicity of **1..*** (see below) that indicates on artefact can be associated with an unlimited number of artefact values.

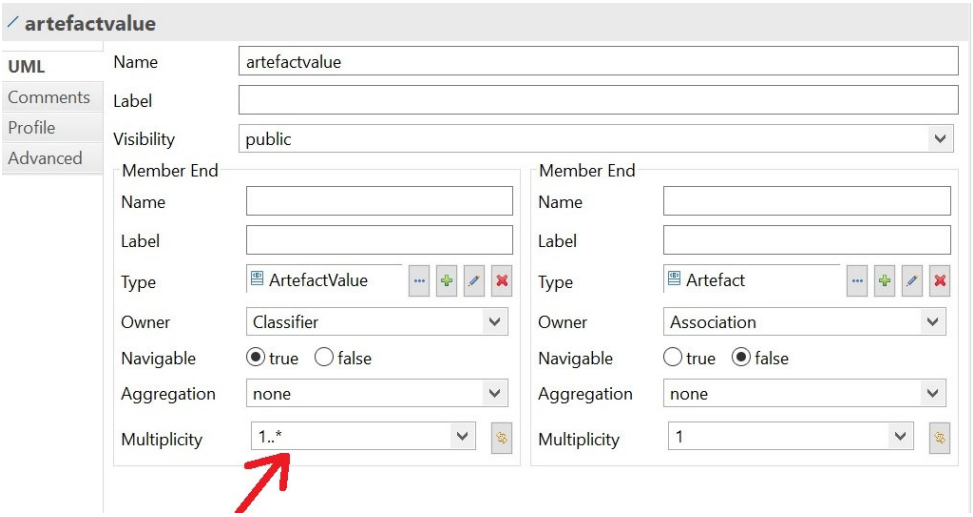

Figure AI.15 Artefact / artefact value link.

AI.4.6 Internal block diagram

The **internal block diagram** captures the internal structure of the generic calibration system. The elements have types listed in the block definition diagram.

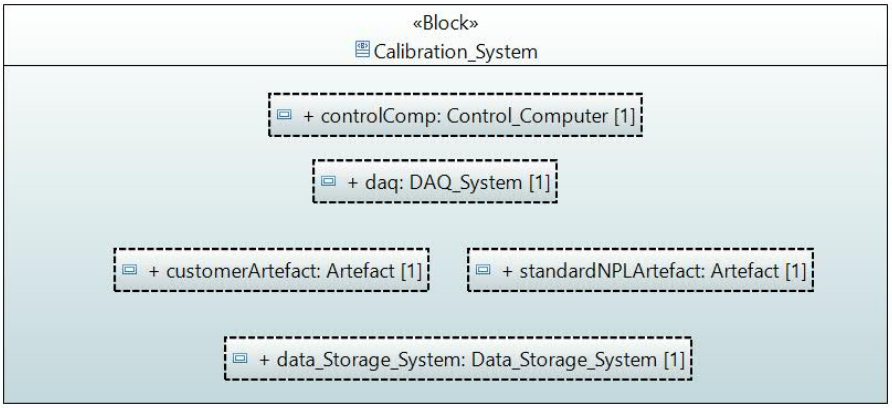

Figure AI.15 Internal block definition diagram.

AI.4.7 Further additions

Features beyond the scope of this version of the model that could be added in a later version include:

- **Ports:** SysML allows ports to be associated with the boundary of a block. They also model the operations that can be carried out and data exchanged via these ports. Ports could be used to specify the interactions between the control computer and the data acquisition and data storage systems.
- **Operations:** The use cases listed in Figure AI.12 provide a good starting point for specifying the operations within the blocks. E.g., the actions linked to the operator: “Power On”, “Place artefacts”, “Log into control PC”, “Commence measurements”, “Monitor measurements”, “Post process”, “Power off” could be associated with **Calibration_System** block. Included use cases could be associated with the blocks associated with Calibration_System, e.g. “Verify username and password” with **Control_Computer**.

APPENDIX II: THE COMPELETE BLOCK DEFINITION DIAGRAM

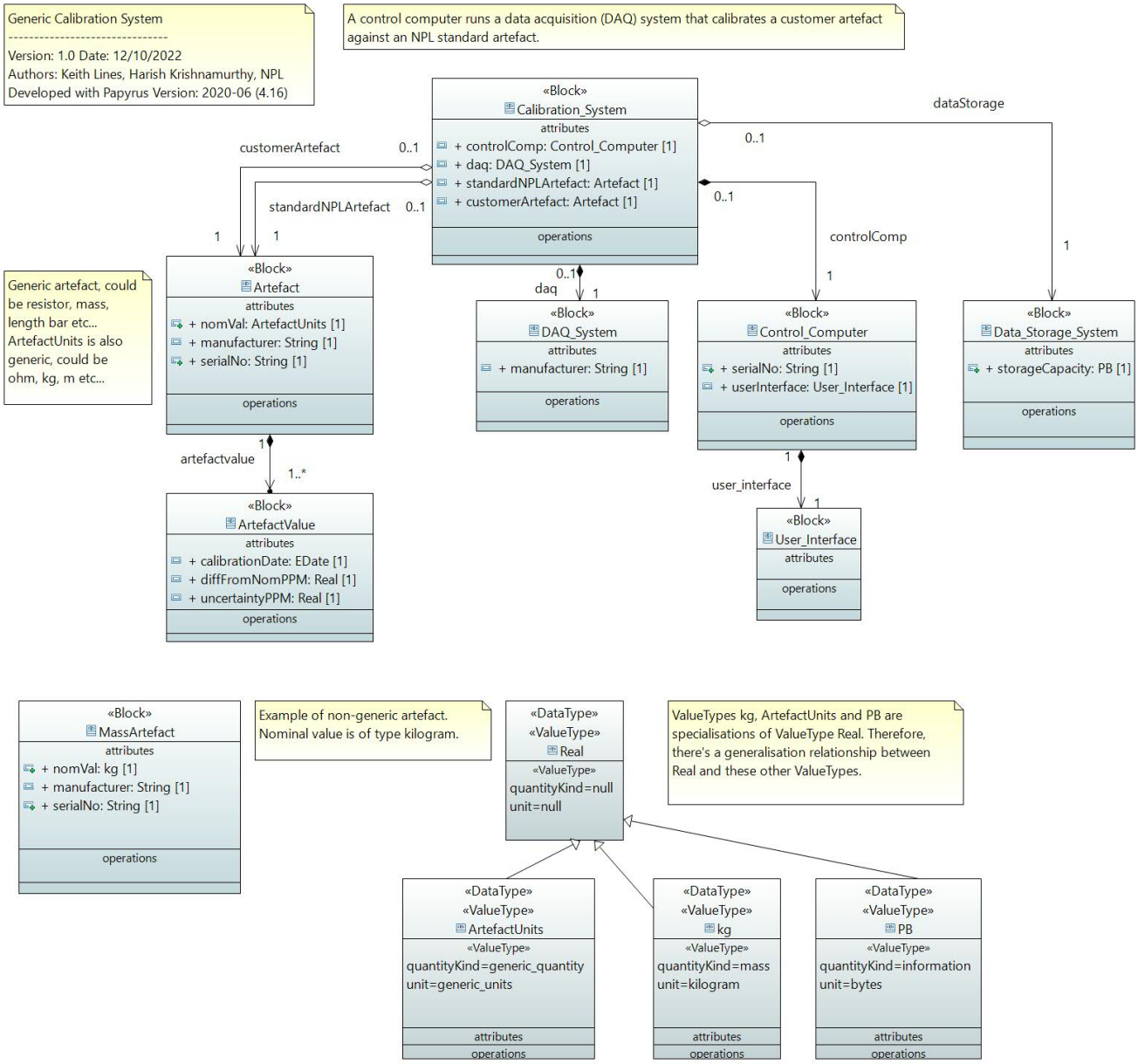

Figure AII.1 Block definition diagram.

APPENDIX III: IMPORTING THE MODEL INTO PAPYRUS

AIII.1 INTRODUCTION

An **archive file** containing the SysML model for the Generic Calibration System is available. The file can be uploaded to the Eclipse Papyrus™ modelling environment. The following appendix provides some guidance on installing Papyrus and uploading the archive file.

AIII.2 INSTALL PAPYRUS

1. The latest version of Papyrus can be downloaded from:

   https://www.eclipse.org/papyrus/download.html

2. Unzip `papyrus-<version>-win64.zip` in an appropriate location, e.g., `c:\eclipse`. Papyrus is be started by clicking:

   `papyrus-<version>-win64\Papyrus\papyrus.exe`

3. When starting Papyrus, an error message beginning `Java was started but returned error code=13 may` appear. If so, Papyrus is using the wrong version of the Java runtime.

4. Papyrus will run using 64-bit Open Java Development Kit 14.0.1. This software can be downloaded from https://jdk.java.net/archive/

5. To install Open Java Development Kit 14.0.1, unzip in an appropriate location, e.g., `c:\openjdk`

6. The `papyrus.ini` file held in the same folder as the Papyrus executable will need to be modified to ensure the correct Java Runtime is being used. An example section of a modified ini file is provided below (modifications indicated in bold):

   ```
   --launcher.appendVmargs
   -vm
   C:/openjdk/openjdk-14.0.2_windows-x64_bin/jdk-14.0.2/bin
   -vmargs
   -Dosgi.requiredJavaVersion=1.8
   ```

7. By default, Papyrus can be used to develop UML projects. Further installation is required for SysML. This section and the following section describe the required installations.

AIII.3 INSTALL ECLIPSE MARKETPLACE

1. After installing Papyrus, **Eclipse Marketplace** is required. Check whether it's been installed by starting up Papyrus and clicking `Help`. If it's been installed, it will be listed. If Eclipse Marketplace is not listed, install as describe below.

2. Click `Help` → `Install New Software…` The `Install` window will now appear.

3. In the `Work with:` drop-down list select `All Available Sites`. Wait for the Name field to populate; this may take a few minutes. Click OK to any error messages that may appear.

4. After the `Name / Version` window has been populated, type `marketplace` into the field just under `Work with:`. Click OK to any error messages that may appear.

5. `General Purpose Tools / **Marketplace Client**` should now be listed. Tick **`Marketplace Client`** and click `Next>` at the bottom of the window.
6. A “review” window will now appear, click `Next>`
7. Accept the terms and conditions (if acceptable). Click `Finish`.
8. Restart Papyrus. Click `Help` and the Eclipse Marketplace should now be available.

AIII.4 INSTALL SYSML 1.4

1. Click Help and select Eclipse Marketplace.
2. Enter SysML into `Find`. To perform the search, select `All Markets` from the drop-down list next to `Find`.
3. Click `install` for the required version of SysML. Read and accept the terms and conditions (if acceptable). Click `Finish`.
4. Restart Papyrus. To check whether SysML is now available, select `File` → `New` → `Papyrus Project`. A New Papyrus Project window will appear. SysML should be an option.

AIII.5 UPLOAD THE ARCHIVE FILE

1. To obtain a copy of the archive file for the case study, contact Keith Lines.
2. It will be assumed that the file has been copied to the C drive of the PC, i.e.

   `C:\SysML\GCS_v1.0.zip`
3. Start Papyrus and create a new project in which to upload the file. Select `File` → `New` → `Papyrus Project`. A New Papyrus Project window will appear. Select SysML 1.4

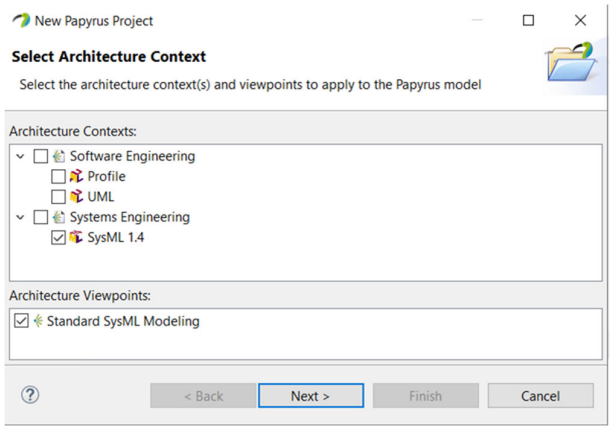

Figure AIII.1 New Papyrus Project window.

4. Select `File` → `Import`. The import window should appear. Select `General` → `Archive File`. Click `Next>`

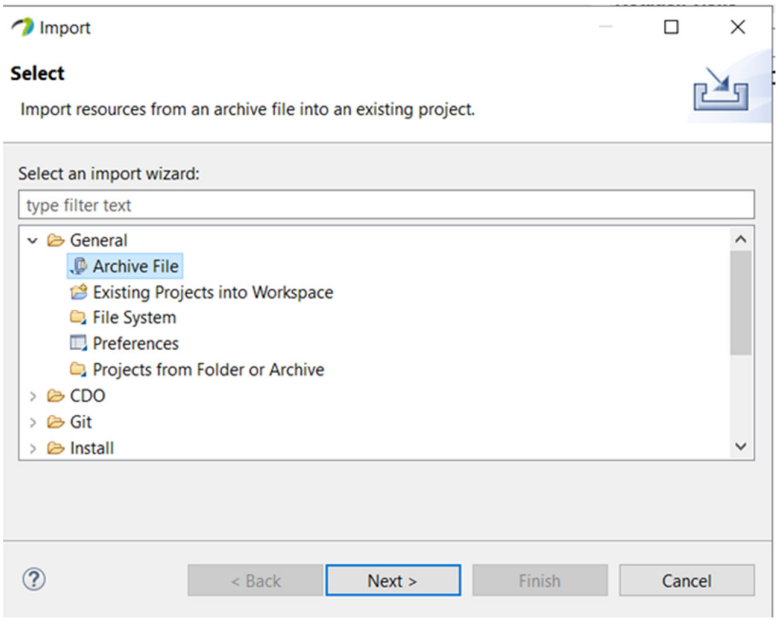

Figure AIII.2 Import wizard window.

5. The `Import` window now appears. Click `Browse…` to select the archive file. Tick the `/`. Enter a folder name in `Into folder`.

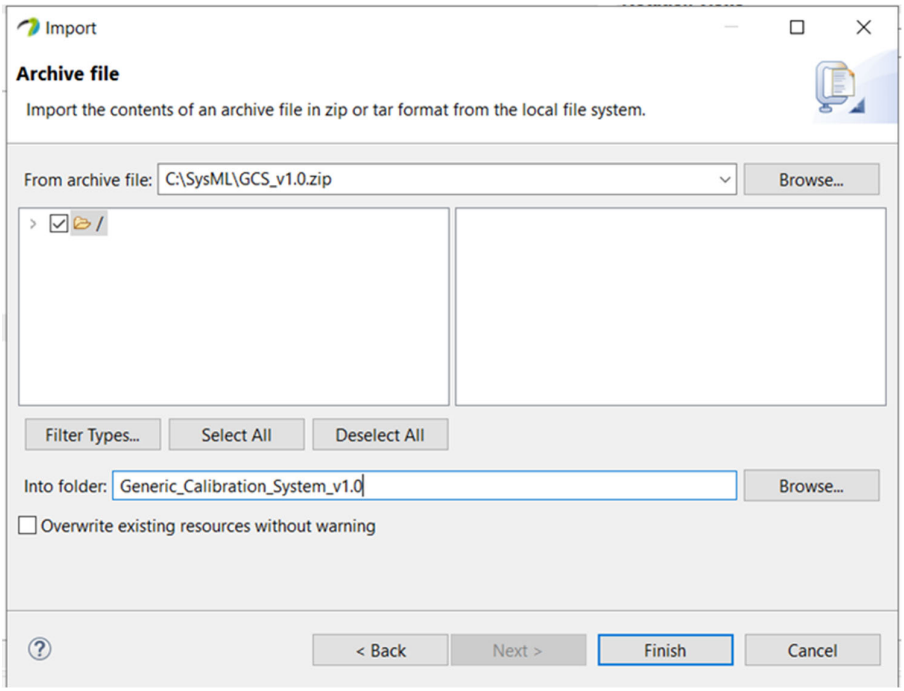

Figure AIII.3 Import window.

6. The project should now upload. In the `Project Explorer` window browse to the model.

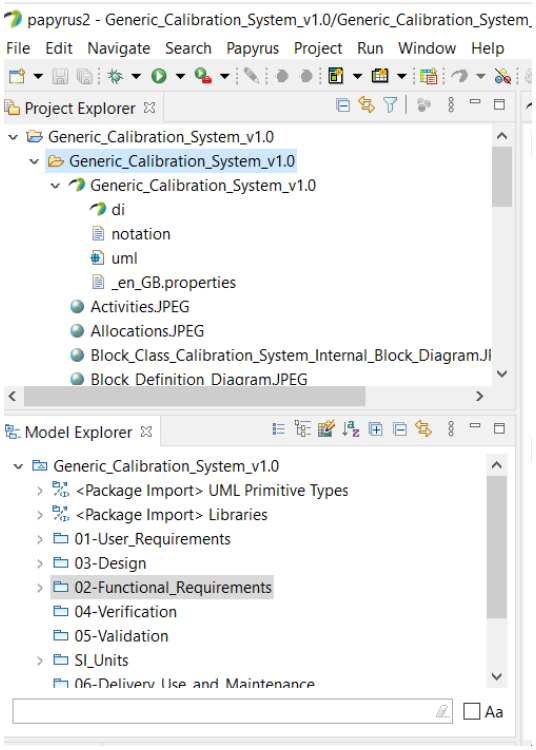

Figure AIII.4 Browse model.